09/828,284
Art Unit: 2192

IN THE CLAIMS:

Amend the claims as follows:

1. (Previously presented) A method of analyzing instructions and data for a program to determine where the instructions and data might result in incorrect results when run on a multiprocessor system, the method comprising the steps of:

dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;

determining which of the instructions and data involve references outside of their domains;

determining which of the references outside of their domains are multiprocessor unsafe references;

generating a report of the multiprocessor unsafe references; and

modifying the instructions and data based on the report.

2. (Previously presented) A method as in claim 1, wherein the instructions and data comprise code that prior to modification is designed for use on a single processor system.

3. (Previously presented) A method as in claim 1, wherein the determining steps, the generating step, and the modifying step are repeated until none of the references are

determined to be multiprocessor unsafe references, whereby the code is modified to be suitable for use on a multiprocessor system.

4. (Previously presented) A method as in claim 1, wherein the instructions and data comprise source code that is analyzed before compilation.

5. (Previously presented) A method as in claim 1, wherein the instructions and data comprise object code that is analyzed before being linked to form an executable.

6. (Previously presented) A method as in claim 1, wherein the instructions and data comprise object code that is analyzed while being linked to form an executable.

7. (Previously presented) A method as in claim 1, wherein the instructions and data comprise interpretable code that is analyzed at run time.

8. (Previously presented) A method as in claim 1, wherein the instructions and data are for execution by a virtual machine, and wherein the instructions and data are analyzed by the virtual machine at run time.

9. (Previously presented) A method as in claim 1, wherein the plural domains include a network domain comprising network functions and data referred to by the network

functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

10. (Previously presented) A method as in claim 9, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

11. (Previously presented) A method as in claim 1, wherein the step of determining which of the references are multiprocessor unsafe further comprises determining which of the references are neither determined to be always multiprocessor safe nor annotated in the code as multiprocessor safe.

12. (Previously presented) A method as in claim 11, wherein references can be annotated in the code as multiprocessor safe because the references switch domains at run time.

13. (Previously presented) A method as in claim 1, wherein the instructions and data are modified by adding annotations that indicate references outside of their domains are multiprocessor safe.

14. (Previously presented) A method as in claim 1, wherein the instructions and data are modified by adding switch domain functions to the instructions and data to change

multiprocessor unsafe references outside of a domain into multiprocessor safe references inside the domain.

15. (Previously presented) A method as in claim 1, wherein the instructions and data are modified automatically based on the report of the multiprocessor unsafe references.

16. (Previously presented) A method as in claim 1, wherein the instructions and data are modified by an expert system aided by a user.

17. (Previously presented) A method as in claim 1, further comprising the steps of:

determining which of the references outside of their domains are purportedly multiprocessor safe references; and

generating a table of the purportedly multiprocessor safe references, the table including the domains to which the references are supposed to refer.

18. (Currently amended) A memory storing information including ~~steps~~ first instructions executable by a processor[[,]] ~~the~~ to perform steps ~~executable~~ to analyze second instructions and data for a program to determine where the second instructions and data might result in incorrect results when run on a multiprocessor system, the steps comprising:

dividing the second instructions and data for the program into plural domains based on symbols used to refer to ~~those~~ the second instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;

determining which of the second instructions and data involve references outside of their domains;

determining which of the references outside of their domains are multiprocessor unsafe references;

generating a report of the multiprocessor unsafe references; and

modifying the second instructions and data based on the report.

19. (Currently amended) A memory as in claim 18, wherein the second instructions and data comprise code that prior to modification is designed for use on a single processor system.

20. (Previously presented) A memory as in claim 18, wherein the determining steps, the generating step, and the modifying step are repeated until none of the references are determined to be multiprocessor unsafe references, whereby the code is modified to be suitable for use on a multiprocessor system.

21. (Currently amended) A memory as in claim 18, wherein the second instructions and data comprise source code that is analyzed before compilation.

22. (Currently amended) A memory as in claim 18, wherein the second instructions and data comprise object code that is analyzed before being linked to form an executable.

23. (Currently amended) A memory as in claim 18, wherein the second instructions and data comprise object code that is analyzed while being linked to form an executable.

24. (Currently amended) A memory as in claim 18, wherein the second instructions and data comprise interpretable code that is analyzed at run time.

25. (Currently amended) A memory as in claim 18, wherein the second instructions and data are for execution by a virtual machine, and wherein the second instructions and data are analyzed by the virtual machine at run time.

26. (Previously presented) A memory as in claim 18, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage

functions, and a general domain comprising other functions and data referred to by the other functions.

27. (Previously presented) A memory as in claim 26, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

28. (Previously presented) A memory as in claim 18, wherein the step of determining which of the references are multiprocessor unsafe further comprises determining which of the references are neither determined to be always multiprocessor safe nor annotated in the code as multiprocessor safe.

29. (Previously presented) A memory as in claim 28, wherein references can be annotated in the code as multiprocessor safe because the references switch domains at run time.

30. (Currently amended) A memory as in claim 18, wherein the second instructions and data are modified by adding annotations that indicate references outside of their domains are multiprocessor safe.

31. (Currently amended) A memory as in claim 18, wherein the second instructions and data are modified by adding switch domain functions to the second instructions

and data to change multiprocessor unsafe references outside of a domain into multiprocessor safe references inside the domain.

32. (Currently amended) A memory as in claim 18, wherein the second instructions and data are modified automatically based on the report of the multiprocessor unsafe references.

33. (Currently amended) A memory as in claim 18, wherein the second instructions and data are modified by an expert system aided by a user.

34. (Previously presented) A memory as in claim 18, wherein the steps further comprise the steps of:

determining which of the references outside of their domains are purportedly multiprocessor safe references; and

generating a table of the purportedly multiprocessor safe references, the table including the domains to which the references are supposed to refer.

35. (Previously presented) A memory as in claim 18, wherein the memory is a removable storage medium, fixed disk, RAM or ROM.

09/828,284
Art Unit: 2192

36. (Currently amended) ~~An~~ A computer system comprising a processor and an analyzer ~~that analyzes~~ to cause the processor to analyze instructions and data for a program to determine where the instructions and data might result in incorrect results when run on a multiprocessor system, the analyzer ~~comprising~~including:

a reference analyzer that divides the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, that determines which of the instructions and data involve references outside of their domains, and that determines which of the references outside of their domains are multiprocessor unsafe references; and

a report generator that generates a report of the multiprocessor unsafe references.

37. (Currently amended) A computer system ~~An analyzer~~ as in claim 36, wherein the analyzer further ~~comprising~~ includes a modifier that modifies the instructions and data based on the report.

38. (Currently amended) A computer system ~~An analyzer~~ as in claim 37, wherein the instructions and data comprise code that prior to modification is designed for use on a single processor system.

39. (Currently amended) A computer system ~~An analyzer~~ as in claim 37, wherein the determining steps, the generating step, and the modifying step are repeated until none of the references are determined to be multiprocessor unsafe references, whereby the code is modified to be suitable for use on a multiprocessor system.

40. (Currently amended) A computer system ~~An analyzer~~ as in claim 37, wherein the instructions and data comprise source code that is analyzed before compilation.

41. (Currently amended) A computer system ~~An analyzer~~ as in claim 37, wherein the instructions and data comprise object code that is analyzed before being linked to form an executable.

42. (Currently amended) A computer system ~~An analyzer~~ as in claim 37, wherein the instructions and data comprise object code that is analyzed while being linked to form an executable.

43. (Currently amended) A computer system ~~An analyzer~~ as in claim 37, wherein the instructions and data comprise interpretable code that is analyzed at run time.

44. (Currently amended) A computer system ~~An analyzer~~ as in claim 37, wherein the instructions and data are for execution by a virtual machine, and wherein the instructions and data are analyzed by the virtual machine at run time.

45. (Currently amended) A computer system ~~An analyzer~~ as in claim 37, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

46. (Currently amended) A computer system ~~An analyzer~~ as in claim 45, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

47. (Currently amended) A computer system ~~An analyzer~~ as in claim 37, wherein determining which of the references are multiprocessor unsafe further comprises determining which of the references are neither determined to be always multiprocessor safe nor annotated in the code as multiprocessor safe.

48. (Currently amended) A computer system ~~An analyzer~~ as in claim 47, wherein references can be annotated in the code as multiprocessor safe because the references switch domains at run time.

49. (Currently amended) A computer system ~~An analyzer~~ as in claim 37, wherein the instructions and data are modified by adding annotations that indicate references outside of their domains are multiprocessor safe.

50. (Currently amended) A computer system ~~An analyzer~~ as in claim 37, wherein the instructions and data are modified by adding switch domain functions to the instructions and data to change multiprocessor unsafe references outside of a domain into multiprocessor safe references inside the domain.

51. (Currently amended) A computer system ~~An analyzer~~ as in claim 37, wherein the instructions and data are modified automatically based on the report of the multiprocessor unsafe references.

52. (Currently amended) A computer system ~~An analyzer~~ as in claim 37, wherein the instructions and data are modified by an expert system aided by a user.

53. (Currently amended) A computer system ~~An analyzer~~ as in claim 37, wherein the reference analyzer further determines which of the references outside of their domains are purportedly multiprocessor safe references; and wherein the analyzer further comprises a table generator that generates a table of the purportedly multiprocessor safe references, the table including the domains to which the references are supposed to refer.

Claims 54 – 79. (Canceled)

80. (Previously presented) A method of analyzing instructions and data and dynamically determining where the instructions and data for a program might result in incorrect results when run on a multiprocessor system, the method comprising the steps of:

dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;

determining which of the instructions and data involve references outside of their domains;

determining which of the references outside of their domains are purportedly multiprocessor safe references;

generating a table of the purportedly multiprocessor safe references, the table including the domains to which the references are supposed to refer;

executing the instructions and data; and

when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, determining if the reference is actually to a domain to which that reference is supposed to refer.

81. (Previously presented) A method as in claim 80, wherein the instructions and data comprise interpretable code.

82. (Previously presented) A method as in claim 80, wherein the instructions and data are for execution by a virtual machine, and wherein the virtual machine performs the method.

83. (Previously presented) A method as in claim 80, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

84. (Previously presented) A method as in claim 83, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

85. (Previously presented) A method as in claim 80, further comprising the steps of:

determining which of the references outside of their domains are multiprocessor unsafe references; and

generating a report of the multiprocessor unsafe references.

86. (Previously presented) A method as in claim 85, further comprising the step of modifying the instructions and data based on the report.

87. (Previously presented) A method as in claim 80, wherein if the reference is not actually to a domain to which that reference is supposed to refer, execution of the instructions and data halts and an error message is generated.

88. (Previously presented) A method as in claim 80, wherein if the reference is not actually to a domain to which that reference is supposed to refer, the instruction or data making the reference is modified.

89. (Previously presented) A method as in claim 88, wherein the instruction or data making the reference is re-executed after being modified.

90. (Currently amended) A memory storing information including ~~steps~~ first instructions executable by a processor[[,]] ~~the~~ to perform steps ~~executable~~ to analyze second instructions and data for a program and dynamically determine where the second instructions and data might result in incorrect results when run on a multiprocessor system, the steps comprising:

dividing the second instructions and data for the program into plural domains based on symbols used to refer to ~~those~~ the second instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;

determining which of the second instructions and data involve references outside of their domains;

determining which of the references outside of their domains are purportedly multiprocessor safe references;

generating a table of the purportedly multiprocessor safe references, the table including the domains to which the references are supposed to refer;

executing the second instructions and data; and

when a reference in the table of purportedly microprocessor safe references is encountered during execution of the second instructions and data, determining if the reference is actually to a domain to which that reference is supposed to refer.

91. (Currently amended) A memory as in claim 90, wherein the second instructions and data comprise interpretable code.

92. (Currently amended) A memory as in claim 90, wherein the second instructions and data are for execution by a virtual machine, and wherein the virtual machine performs the steps.

93. (Previously presented) A memory as in claim 90, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

94. (Previously presented) A memory as in claim 93, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

95. (Currently amended) A memory as in claim 90, wherein the first instructions further comprising comprise instructions to cause the processor to perform the steps of:

determining which of the references outside of their domains are multiprocessor unsafe references; and

generating a report of the multiprocessor unsafe references.

96. (Currently amended) A memory as in claim 95, further comprising the step of modifying the second instructions and data based on the report.

97. (Currently amended) A memory as in claim 90, wherein if the reference is not actually to a domain to which that reference is supposed to refer, execution of the second instructions and data halts and an error message is generated.

98. (Previously presented) A memory as in claim 90, wherein if the reference is not actually to a domain to which that reference is supposed to refer, the instruction or data making the reference is modified.

99. (Previously presented) A memory as in claim 98, wherein the instruction or data making the reference is re-executed after being modified.

100. (Currently amended) A computer system for analyzing instructions and data for a program and dynamically determining where the instructions and data might result in incorrect results when run on a multiprocessor system, the computer system comprising a processor and, executable by the processor:

a reference analyzer that divides the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, that determines which of the instructions and data involve references outside of their domains, and that determines which of the references outside of their domains are purportedly multiprocessor safe references;

a table generator that generates a table of the purportedly multiprocessor safe references, the table including the domains to which the references are supposed to refer;

a reference tracker that tracks references made by the instructions and data; and

a comparator that determines, when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, if the reference is actually to a domain to which that reference is supposed to refer.

101. (Currently amended) A computer system as in claim 100, wherein the instructions and data comprise interpretable code.

102. (Currently amended) A computer system as in claim 100, wherein the instructions and data are for execution by a virtual machine, and wherein the virtual machine implements the system.

103. (Currently amended) A computer system as in claim 100, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

104. (Currently amended) A computer system as in claim 103, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

105. (Currently amended) A computer system as in claim 100, wherein the reference analyzer further determines which of the references outside of their domains are multiprocessor unsafe references, and wherein the system further comprises a report generator that generates a report of the multiprocessor unsafe references.

106. (Currently amended) A computer system as in claim 105, further comprising a modifier that modifies the instructions and data based on the report.

107. (Currently amended) A computer system as in claim 100, wherein if the reference is not actually to a domain to which that reference is supposed to refer, execution of the instructions and data halts and an error message is generated.

108. (Currently amended) A computer system as in claim 100, further comprising a modifier that, if the reference is not actually to a domain to which that reference is supposed to refer, modifies the instruction or data making the reference.

109. (Currently amended) A computer system as in claim 108, wherein the instruction or data making the reference is re-executed after being modified.

110. (Previously presented) A method of analyzing instructions and data for a program to determine where the instructions and data might result in incorrect results when run on a system having multiple resources of a type used concurrently, the method comprising the steps of:

dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the system configured to use at most one of the resources at a time to execute instructions and to access data from any one domain;

determining which of the instructions and data involve references outside of their domains;

determining which of the references outside of their domains are unsafe references;

generating a report of the unsafe references; and

modifying the instructions and data based on the report.

Claims 111 - 125. (Canceled)

126. (Previously presented) A method of dynamically analyzing instructions and data for a program to determine where the instructions and data result in domain violations when run on a multiprocessor system, the method comprising the steps of:

dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;

accessing a table of purportedly microprocessor safe references by the instructions and data outside of their domains, the table including the domains to which the references are supposed to refer;

executing the instructions and data; and

when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, determining if the reference is actually to a domain to which that reference is supposed to refer.

127. (Previously presented) A method as in claim 126, wherein the instructions and data comprise interpretable code.

128. (Previously presented) A method as in claim 126, wherein the instructions and data are for execution by a virtual machine, and wherein the virtual machine performs the method.

129. (Previously presented) A method as in claim 126, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

130. (Previously presented) A method as in claim 129, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

131. (Previously presented) A method as in claim 126, wherein if the reference is not actually to a domain to which that reference is supposed to refer, execution of the instructions and data halts and an error message is generated.

132. (Previously presented) A method as in claim 126, wherein if the reference is not actually to a domain to which that reference is supposed to refer, the instruction or data making the reference is modified.

133. (Previously presented) A method as in claim 132, wherein the instruction or data making the reference is re-executed after being modified.

134. (Previously presented) A memory storing information including steps executable by a processor, the steps executable to dynamically analyze instructions and data for a program to determine where the instructions and data result in domain violations when run on a multiprocessor system, the steps comprising:

dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;

accessing a table of purportedly microprocessor safe references by the instructions and data outside of their domains, the table including the domains to which the references are supposed to refer;

executing the instructions and data; and

when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, determining if the reference is actually to a domain to which that reference is supposed to refer.

135. (Previously presented) A memory as in claim 134, wherein the instructions and data comprise interpretable code.

136. (Previously presented) A memory as in claim 134, wherein the instructions and data are for execution by a virtual machine, and wherein the virtual machine performs the steps.

137. (Previously presented) A memory as in claim 134, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

138. (Previously presented) A memory as in claim 137, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

139. (Previously presented) A memory as in claim 134, wherein if the reference is not actually to a domain to which that reference is supposed to refer, execution of the instructions and data halts and an error message is generated.

140. (Previously presented) A memory as in claim 134, wherein if the reference is not actually to a domain to which that reference is supposed to refer, the instruction or data making the reference is modified.

141. (Previously presented) A memory as in claim 140, wherein the instruction or data making the reference is re-executed after being modified.

142. (Currently amended) A computer system comprising a processor and a checker ~~that~~ to cause the processor to dynamically ~~analyzes~~ analyze instructions and data for a program to determine where the instructions and data result in domain violations when run on a multiprocessor system, the checker comprising:

an analyzer that divides the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain;

an interface to a table of purportedly microprocessor safe references by the instructions and data outside of their domains, the table including the domains to which the references are supposed to refer; and

a reference tracker that tracks references made by the instructions and data; and

a comparator that determines, when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, if the reference is actually to a domain to which that reference is supposed to refer.

143. (Currently amended) A computer system ~~checker~~ as in claim 142, wherein the instructions and data comprise interpretable code.

144. (Currently amended) A computer system ~~checker~~ as in claim 142, wherein the instructions and data are for execution by a virtual machine, and wherein the virtual machine includes the checker.

145. (Currently amended) A computer system ~~checker~~ as in claim 142, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

146. (Currently amended) A computer system ~~checker~~ as in claim 145, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

147. (Currently amended) A computer system ~~checker~~ as in claim 142, wherein if the reference is not actually to a domain to which that reference is supposed to refer, execution of the instructions and data halts and an error message is generated.

148. (Currently amended) A computer system ~~checker~~ as in claim 142, further comprising a modifier that, if the reference is not actually to a domain to which that reference is supposed to refer, modifies the instruction or data making the reference.

149. (Currently amended) A computer system ~~checker~~ as in claim 148, wherein the instruction or data making the reference is re-executed after being modified.

150. (Currently amended) A computer system ~~checker~~ as in claim 142, wherein the checker is embodied as a function in the instructions and data.

151. (Currently amended) A computer system ~~checker~~ as in claim 142, wherein the checker runs concurrently with execution of the instructions and data.

152. (Previously presented) A method of dynamically analyzing instructions and data for a program to determine where the instructions and data result in domain violations when run on a system having multiple resources of a type used concurrently, the method comprising the steps of:

dividing the instructions and data for the program into plural domains based on symbols used to refer to those instructions and data, the system configured to use at most one of the resources at a time to execute instructions and to access data from any one domain;

accessing a table of purportedly safe references by the instructions and data outside of their domains, the table including the domains to which the references are supposed to refer;

executing the instructions and data; and

when a reference in the table of purportedly safe references is encountered during execution of the instructions and data, determining if the reference is actually to a domain to which that reference is supposed to refer.

153 - 164. (Canceled)